Defining a baseline complexity model for ERP Systems over SaaS

Kenneth J. Spiteri
Cristina Luca
Tim Reynolds
George Wilson
Dept. Computing and Technology
Anglia Ruskin University
Cambridge, UK
Kenneth.spiteri@student.anglia.ac.uk
Cristina.luca@anglia.ac.uk
Tim.reynolds@anglia.ac.uk
George.wilson@anglia.ac.uk

***Abstract*—Our research investigates applying the Software as a Service (SaaS) model for hosted applications to more complex business systems such as an Enterprise Resource Management System (ERP). This application of the service model is still in its infancy and we present some challenges to the technology. We will initially be defining a measure of complexity for business systems and applying this as a baseline to complex business systems within a pure SaaS model on the Cloud, considering the elements making up SaaS and inter-relating these with this definition of business complexity. In this research we will be applying elements of Complex Systems Theory, Network Complexity Theory and Programmatic Complexity Models, and extending these elements for our own application to this service model within this complex business context.**

Keywords-Complex Systems Theory; Business Systems; The Cloud; Network Complexity Theory, Programmatic Complexity; ERP; SaaS

I. INTRODUCTION

Cloud computing, is a representation of the Internet for the delivery of four distinct hosted service models over the distributed network. The context of this research will be primarily in respect to the model for distribution of software applications hosted and managed by a service provider. This is defined as Software as a Service (SaaS). The key differentiator for this model is that in SaaS the service provider supplies the software product beside the hardware infrastructure, and the service consumer only requires an Internet enabled device to access the software service (Chappell, 2008). With the significant innovations in distributed computing as well as improved access to high-speed Internet, there has been an increase in the feasibility and interest in this form of cloud computing (Wittmann, 2010). As major industry players have started to invest heavily in infrastructures to support cloud computing (e.g. datacentres) and with the introduction of faster mobile network connectivity (especially 4G), cloud services are becoming more accessible and economical. This has seen SaaS becoming increasingly popular amongst business users and consumers. Other reasons for this popularity include cost efficiency, both in currency and carbon, balance of capital vs. recurrent costs, skills availability and ease of use. Cloud services including SaaS are therefore increasingly seen as a sustainable technology.

Whilst the SaaS model has generated much business interest, to date we find its application having been largely confined to flat business services. A flat service, as we define it, is that of a linear model tracking existing simplified business processes. These linear activities are generally tangible excluding complex business systems due to the associated greater technical, business and legal issues (Perrone et al, 2010; Sarrell, 2010). Our current research examines and defines the potential constraints within the existing SaaS model when applied to complex business systems. Whilst the definition of complexity varies widely depending on the science that it is applied to, in this research we will define software complexity within a business context for the SaaS model. The Enterprise Resource Planning Systems (ERP) will be used as an example of what could be considered a complex business system. Defining a measure of complexity would then enable us to develop a deployment framework (Spiteri et al, 2012) as a benchmark for the feasibility of complex systems on the Cloud.

II. CLOUD COMPUTING AND SAAS

The term 'Cloud Computing' was devised as an abstraction of the Internet and the technological infrastructure that supports it (HIPAA, 2009). It is therefore a loose metaphor for the Internet viewed as a medium to host a variety of services. These hosted services cover four main models: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Communication-as-a-Service (CaaS) and Software-as-a-Service (SaaS).

In the IaaS service model, an organisation outsources the hardware used to support its operations (e.g. storage, servers and networking components) to a third party service provider. This service provider in turn operates the equipment and is responsible for its management. In contrast the PaaS model uses virtualised servers to provide operating system and application services and is considered primarily as a development platform.

The CaaS model provides organisations with the facility to outsource their communication requirements through such mediums as Voice over IP, Voice Conferencing and TelePresence.

The SaaS service model is a key differentiator from the other services in that the service provider supplies the software product besides the hardware infrastructure. The provider interacts with the service consumer through various web front-ends that include a myriad of devices such as laptops, mobiles, tablets and most other web enabled devices. SaaS therefore supports a wide-ranging market where services can be wide ranging from Web-based email, stock control, database processing to underwriting algorithms.

Our study primarily considers the SaaS model over the other three cloud services. Since the service provider hosts both the application and the data, the consumer is able to use the service from any location with Internet access. For this reason SaaS is also referred to as software 'on-demand', where instead of installing software on a local computer or business network, the software service required is stored and provided online 'within the cloud'.

III. OPERATIONAL CONSIDERATIONS

From an operational point of view applicable to all the four types of service, a cloud computing application presents unique requirements that a service provider must meet (Kimberling, 2008; Force.com, 2009). These could be described as:

Operations/Availability – an application should be built, hosted and maintained by the provider such that it remains available for users.

Deployment – an application should be "on-demand" i.e. it must have automated mechanisms that let users sign up, log in, and start work with limited latency.

Elasticity – an application must be elastic, automatically scaling its consumption of computing resources to the demands placed on it at any given time. Shared, cloud-based applications must also address some unique technical requirements.

Customisation – an application must enable each organisation to customise its data model, interface, and business logic to meet its internal requirements and needs.

Security – an application must have strong, configurable security mechanisms that enable an organisation to secure data among different types of users both emulating an intranet and/or extranet.

Upgradeability – an application's code base must be easy to upgrade and update, without compromising organisation-specific customisations and configurations.

Integration – an application must be able to combine selected data and functionality via industry-standard protocols, both interoperable with other cloud applications and traditional on-premises and legacy applications.

Device Independence – an application must work on various devices, including desktop computers and mobile devices, so users can be productive wherever and however they work.

IV. AN EXAMPLE OF A COMPLEX BUSINESS SYSTEM

The latest incarnation of Enterprise Resource Planning Systems (ERP) can be considered as tracking complex business systems. The traditional view of an ERP referred to enterprise wide software for larger organisations; its modular features generally allow planning in the use of organisational wide resources, and these systems are thus mostly used in larger more industrial types of businesses. The use of ERP has however changed (Tech-Faq, 2009; Web Based ERP, 2006) and has over the years been applied to a very comprehensive range of business types, irrespective of the size of industry or organisation. This has been facilitated by the modular nature and innovation of ERP vendors and systems that enable selective deployment levels, matching complexity of the software itself to that of the organisation. ERP systems can cover a wide range of functions often deployed as modules, examples of which include the management of Human Resources, Supply Chain, Customer Relations, Finance, Manufacturing, Warehouse and Inventory Management, amongst others (Welch and Kordysh, 2007). Most proprietary ERP systems have adopted this modular architecture. Previously all these functions within the business were managed through their own disparate applications where each of these departments would typically have their own specialist software system optimised for each particular role. However these individual applications can now be unified and integrated within the ERP system. An ERP therefore is a mechanism to integrate enterprise data and processes within one single system, a software architecture that serves the needs of users in finance as it does for those in human resources or in the warehouse. A second tier distribution channel has become established where ERP applications have been tailored to specialise in specific market sectors, for example servicing the fresh food processing industry, automotive or retail businesses, etc.

ERP systems will also usually have many components including hardware and software in order to achieve this integration (Hvolby and Trienekens, 2010; Tech-Faq, 2009). Most ERP systems use a unified database to store data for the various functions throughout the organisation, such that the various departments can more easily share and communicate information. This integrated approach can have a tremendous benefit with respect to meeting market requirements and dealing with the challenges for companies in the global knowledge economy (Wailgum, 2010).

V. COMPLEXITY IN BUSINESS AND THE ERP SYSTEM

There has been considerable research on the subject of complex systems, with complexity theory being applied to a wide range of sciences, technology and management, and has become a major field of interdisciplinary research (Nicolis, 2009; Mikulecky, 2001).

In this respect, the definition of complexity varies widely depending on the science that it is being applied to. Some examples would be algorithm complexity defined in

computer science, logistical complexity within operations management and organism complexity within biological sciences. As researchers like Amaral and Uzzi (2007) observed, there is evidence of complexity in many of these disparate systems where it was also noted that on simplifying complexity concepts, generic aspects of these apply across a wide-spectrum of research fields. This in turn has led to an understanding, both qualitative and quantitative, of the complex systems encountered in nature, technology and everyday systems.

Based on this understanding of complexity, we propose the application of three key theories of Complex System Theory, Programmatic complexity and Network complexity, to ERP business systems, with the aim of defining a baseline complexity model against which our research framework (Spiteri et al, 2012) can be validated.

Complex Systems Theory

Complexity theory is the study of complex, nonlinear, dynamic systems where the resultant output is of a simple nature (Levy, 2000). This is the distinguishing factor from chaos theory, of which primary concern is with systems in which the recursive application of nonlinear deterministic functions can give rise to apparent random behaviour and subtle patterns. In this case, a simple system can give rise to complexity in its output, and as such the theory is focused more on the output than on the system itself.

This systems approach led to the creation of a variety of definitions for Complex Systems. These systems are described as being *complex* because they have numerous internal elements and *dynamic* because their global behaviour is governed by local interactions between their elements (Geyer et al, 2005, Richardson, 2004).

Organised vs. dis-organised complexity

The discovery of common abstractions and mechanisms greatly facilitates our understanding of complex systems (Edmonds, 1999; Young et al, 2007). For example, in an organisation where a service is provided, a product is manufactured, material is purchased, wages are paid, statuary financials are reported etc., recognising the properties common to all such organisations allows us to more easily validate concepts across such entities.

Researchers have noted that systems do not express a single hierarchy but rather many different hierarchies are usually present within the same complex system (Simon, 1973; Richardson, 2004). In our view this concept can be applied to a business example by breaking down an organisation into its different hierarchies, such as its Purchasing department, Production department or Finance department. This decomposition of the whole represents a structural hierarchy, which is part of a wider hierarchy. It is essential to view a system from both perspectives, understanding the whole hierarchy structure as well as its sub hierarchies Skyttner (2001).

An organised complexity model proposes a view that there exists a hierarchy of levels of organisation, with each level more complex than the underlying one. A level will be representative of emerging elements which do not exist at the lower levels. In our extension of this concept to the ERP system model, we could consider an example where say within a Purchase Order Processing hierarchy, a Product Receipt emerges from

Purchase document Lines indicating Product, Units and Prices which in turn emerges from a Purchase document header indicating Supplier details.

In analysing a complex software system such as an ERP, this would likely lead to identification of multiple elements that must interact in a multitude of intricate ways. In some cases with little definite commonality among either the parts or their interactions; this is an example of disorganised complexity (Checkland, 1981). In modelling an ERP complexity, the organised elements in conjunction with the dis-organised ones together within one whole have to be considered.

Tangible vs. Intangible Complexity

Our model of complexity, as applied to an ERP system, has to also consider elements within the operational environment of the system for more accurate modelling because this could have a direct impact on the overall complexity of the system and required framework. To this end we factor-in complexities that can be considered as a tangible or of an intangible nature. Tangible complexity can be identified as industry complexity, regulatory complexity, organisational complexity, and process complexity. Intangible complexities would relate to scenarios such as Monte-Carlo simulations, unintended consequences, correction of errors, sustainability, expansion potential and maintenance management.

As organisations experience growth and scale, the tendency for complexity within the business increases. This frequently results from management decisions to adapt to internal and external pressures, and due to externalities or surrounding market forces (Kimberling, 2010; Jagersma and Klaas, 2004). It is quite likely this complexity would in turn translate into the IT systems deployed to manage the organisation itself. Highly complex processes are error-prone, difficult to understand and difficult to maintain (Muketha et al, 2010), and this is reflected in the systems implemented to manage them.

Programmatic Complexity

Since an ERP system is inherently a software system we need to consider the impact of programmatic complexity that the ERP model would be defined by. Programmatic complexity is in essence the numerous elements defining the software and their interactions. As the number of elements increases, we see the interactions between them increasing exponentially (Kearney et al, 1986; Megiddo, 1987).

If we consider Data Access Complexity, when an element within the hierarchy receives a message, it acts upon it as defined by functions inside the element (Card and Agresti, 1988). As elements are arranged together within a business process or system process reflecting it, then the definition of its communication access points allows other elements to be exposed and to send them messages.

Dataflow complexity, another subdivision of Programmatic complexity, is defined by the functional elements within the software hierarchy connected by directed dataflow queues (Falgout, 2011; McCabe, 1976). The dataflow queues transport the data between connected functions, with the output of one functional element being the input of another. Within our ERP process this could translate to distinct functions manipulating data within the hierarchy levels. In this case, the consolidated data inputs and outputs result in the

general inputs and outputs of the hierarchical levels and elements making up the process nodes.

Network Complexity

As we apply the models of Systems Theory and Programmatic Complexity to an ERP System Model, the concepts of interactions between hierarchies and elements could also be described through interrelationships of functional nodes within a network, where the state of each node is the function of its connections to other nodes. This is where we see Network Complexity theory used in defining this behavioural model.

Network theory is however more interested in the emergent order and patterns in complex systems than in an attempt to find a simple mathematical "engine" in the system (Levy, 2000; Wolfram, 1985). Network models often try to capture the essence of interaction among the many elements in a system, by modelling large numbers of nodes connected by simple logical rules.

Baseline Complexity Model for Complex Business Systems

As highlighted in the complexity theories and models described above, a common definition can be inferred that complexity is the expression of numerous elements in a system where these elements are interrelated (Lucas, 2006; Amaral and Uzzi, 2007). This is in contrast to simple systems which would therefore have a small number of well understood elements. In our application of these rules to the business and ERP system model, we define baseline complexity as a measure of tangible complexity i.e. a composite of business and technological elements determined by factors such as the number of sub-processes as well as their interactions, interdependencies and relationships within the process environment. When these features are applied to a business, our definition would cover structure and interdependence of business elements such as:

Finance - for book-keeping and financial transaction recording, enabling audit and the creation of management accounting reports.

Sales Order Processing - which facilitates and records sales of product or services and related transactions, including related document transactions and reporting.

Purchase Order Processing - which facilitates and records purchases of product or services and related transactions, including related document printouts and reporting.

Warehouse and Inventory Management - to ensure accurate stock management, which is usually a key element for accurate business flow within other elements of the ERP, enabling more efficient use of product inventories, control of picking locations, shelf live and storage.

Manufacturing - allows production companies to manage their bill of materials, production process routings, production scheduling and recoding product consumption and output including waste and production costs.

Human Resources - facilitates management of employee data, payment of skills quotients, salaries, vacation and absence records.

Services Management – allows service companies to manage the provision of their services to customers including the handling of warranty claims.

Customer Relationship Management - allows management of customers and contacts, both vertical and horizontal sales information yielding data relevant to the marketing teams.

Transport Logistics – management of the business transport requirements in timely distribution of product from suppliers to customers, both for business owning their own fleets or third party transport. This allows for more efficient planning and control of carbon costs of transportation directly within the system business process.

Quality Control – Management and recording of quality assurance and controls for product and related processes ensuring that the business facilities are operating within appropriate quality conditions and ensuring fitness for purpose of the product..

Sales and Purchase Forecasting – as a tool for planning both current and future, supply and demand, for the product.

These business elements and their processes are consistently found across organisations and therefore tend to also be reflected within ERP systems as default modules integrated within the application.

We find an ERP implementation might not necessarily imply a complex business system in itself. Usually partial modules can be implemented separately to cover only particular aspects of the business. Our research therefore suggests that a complex business process within the context of an ERP is one that has most of these elements deployed as part of the general business processes of its trading environment. We propose that the more inter-related processes and services an organisation has, the more likely that complexity will present issues to the integration of the ERP within its business structures.

Thus our definition of a complex business system is therefore one that covers many of these processes operating inter-dependently within its functions. Such a system would fall within the characteristics of what could be considered a fully-fledged ERP system implementation. We also find that system complexity would increase even more if disparate systems and technologies were used to handle the various business elements, as this would traditionally require data interfaces, mapping structures and communication protocols, with some of these potentially being provided as services through third parties having their own proprietary systems.

VI. CHALLENGES FOR THE SAAS MODEL

Our development of such a framework (Spiteri et al, 2012) for SaaS has to therefore consider the elements and possible constraints that this delivery model has to function within, irrespective of whether boundaries are tangible and technological, or simply perceived by the potential consumers of the service. These constraints initiate early in the deployment process, with the service provider delivery consideration requiring a shift over traditional delivery models (Leon, 2010). The development life cycle will require a shorter continuous development stream, supported with a critical in-depth software testing process to ensure there are no negative impacts to end-customers' daily operations.

The establishment costs for a SaaS service provider would also require a larger investment to cover the initial effort for development of the core systems needed to support the requirements of a broad customer base. Setup of operations would also need a wider range of resource skills and specialised knowledge. The responsibility for the

hosting of the software is transferred to the service provider, ensuring a consistent provision of the service. Resilient networking infrastructure, including systems and network monitoring facilities and data centre facilities, increase start-up costs considerably. Compounding this, initial revenue generated from the provision of SaaS services is likely to be lower in the shorter term, and revenue growth has to be expected at a slower rate than traditional delivery models where customers would pay software license fees up-front. As an on-going process, agility in the identification of reported issues, customer needs and feedback, is key, and this implies frequent software releases and upgrades (Veverka, 2010; Holme et al, 2008).

Our research also suggests that development would need to consider the entire SaaS platform, which depending on the complexity of the software solution would require a step-up in customer support, increased services in training, and considerations of the impact on customers' business following system changes. With these barriers to entry, for consumers this would likely entail a reduced selection in their options for solution providers.

Further research will explore how standard mechanisms for reducing complexity might still apply to our complex business system model. As outlined by Behringer (2009) and Moody (2000), concepts to be considered include:

Divide and Conquer: In this context this would entail the business system disaggregate into smaller more manageable processes and functions.

Shift or hide complexity: Elements at all levels will be translated simply to their inputs and outputs, abstracting the complexity within the element themselves.

Defining Meta-Languages: Providing simplified meta-languages, hiding complexity in simplified commands. Examples of such meta-languages are the Common Information Model (CIM), Routing Policy Specification Language (RPDSL), or Ecommerce Extensible Mark-up Language (ecXML) (Spiteri, 2004).

Structural Approach: This is a mechanism to search analytically for dependencies and try to discover ways to reduce them. In business these could be dependencies between the process defined elements.

We believe that SaaS service providers would need to make considerations for reducing the overall complexity of their solution and take these into account when defining a SaaS model for their IT business solution. As suggested by Holme et al, (2008), proposed elements that might have a bearing on this when current technology is factored-in include:

- The requirement for a quick implementation over a longer traditional deployment model.
- Importance of low start-up costs and initial investment on IT systems.
- The nature of the business processes and the distribution of the workforce in consuming business data.
- Level of Integration with other systems.
- Commonality of the business processes and therefore level of customisation an IT system would otherwise require.

- Time critical nature of applications.
- Data generation sources and data distribution existing outside firewalls.

VII. MODELLING A COMPLEX BUSINESS SYSTEM

The authors' current work is an investigation into techniques to model a complex business system (with most of the elements of an ERP) as a pure SaaS on the Cloud. It relates the elements of a SaaS system with the concepts of complexity as outlined in this work. Whilst the emergence of business systems on the Cloud is on the increase (Hall, 2010) and certainly many vendors are using ERP sales rhetoric for these online solutions, organisations could find that these do not yet cover the complexity of a full ERP implementation in respect to the definition and criteria cited here. In particular the simplification of elements, where business systems already making a presence on the Cloud (e.g. customer relationship management functions, inventory management, finance) do not fit the unified features of a traditional complex ERP.

We consider some online business solutions marketed for the Cloud on the other hand as more likely to fall under the PaaS model, where in effect it is the hosting of the software which is online but other elements of the deployment model remain within traditional deployment standards. Important research questions that we will be addressing include:

- What elements define a hosted service and how could this definition be applied to complex business systems?
- What defines a complex business system, and at what level would an ERP implementation be classified as complex?
- What technological benefits and constraints can be identified for hosting complex business system services considering current technological innovations?
- What is the view of businesses that have adopted or likely to adopt an ERP in accepting a hosted service?
- What are the significant issues, technological or otherwise, for the application of SaaS to ERP?

Our approach will be most relevant for businesses that have adopted or are likely to adopt a traditional ERP system in formulating an assessment of the factors that should be considered in the utilisation of the SaaS service model.

VIII. SUMMARY AND FURTHER WORK

The development of a framework to model complex business systems as a SaaS model on the Cloud (Spiteri et al, 2012) is proposed based on a definition of complexity applied at an organisational process level in relation to a software engineering context. This work should identify high priority issues possibly restricting the adoption of hosted internet services in complex business environments (whether real or perceived) and hence indicating a focus for future research in mitigating these issues. Resolution of such issues are likely to be i) hardware-oriented, in development of new technology, ii) education-oriented, in effectively communicating the technology, and iii) economics-oriented, in establishing to what extent cloud technologies will be sustainable and cost effective over

differing time frames. In addition this would assist service providers with a strategic view in their generation of a road map for the technology.

REFERENCES

[1] Chappell, D. (2008), “A Short Introduction to Cloud Platforms”, David Chappell & Associates, [Online] www.davidchappell.com/CloudPlatforms--Chappell.pdf, (Access Date May 2009)

[2] Hall, K. (2010), "Gartner: SaaS sales will grow 16.2% to $10.7bn in 2011". ComputerWeekly.com. Reed Business Information Ltd., [Online] http://www.computerweekly.com/Articles/2010/12/14/244489/Gartner-SaaS-sales-will-grow-16.2-to-10.7bn-in-2011.htm. (Access Date 8 July 2011).

[3] HIPAA, (2009), ”What is cloud computing? Name is symbolic, services are real”, Briefings on HIPAA, v9 i12, pp10-11, 2009.

[4] Hvolby, H.H. and Trienekens, J.H. (2010), “Challenges in Business System Integration.” Computers in Industry, 61 (9): 808-814.

[5] Nicolis, G. and Nicolis, C. (2009), “Foundations of Complex Systems”, European Review, Vol. 17, Issue 02, pp 237-248

[6] Kimberling, E. (2008), “SaaS vs. Traditional ERP: Five Key Differentiators,” Panorama Consulting Group, [Online], http://panorama-consulting.com/saas-vs-traditional-erp-five-key-differentiators/, (Access Date Sep 2010)

[7] Kimberling, E. (2010), “SaaS, On Premise, Cloud, Best of Breed: Making Sense of All the ERP System Options”, Panorama Consulting Group, 2010, http://panorama-consulting.com/saas-on-premise-cloud-best-of-breed-making-sense-of-all-the-erp-system-options/, (Access Date Oct 2010).

[8] Leon, J.F. (2010), “Vetting a Vendor: Questions to Ask Before Making an Investment”, Journal of Accountancy, [Online], http://www.journalofaccountancy.com/Issues/2010/Oct/VettingaVendor.htm, (Access Date Sep 2010)

[9] Lucas, C. (2006), “Quantifying Complexity Theory”, [Online], http://www.calresco.org/lucas/quantify.htm, (Access Date Sept 2011)

[10] Tech-Faq, (2009), “What is ERP?”, Topbits Tech Community, [Online], http://www.tech-faq.com/erp.shtml, (Access Date Sep 2010)

[11] Veverka, M. (2010), “A Private Party”, Barron’s Cover, Barron’s, [Online], http://online.barrons.com/article, (Access Date Oct 2010)

[12] Wailgum, T. (2010), “ERP (Enterprise Resource Planning) topics covering definition, objectives, systems and solutions”, [Online], http://www.cio.com/article/40323/ERP_Definition_and_Solutions, (Access Date Sept 2011)

[13] Web Based ERP, (2006), “Web Based ERP Software”, [Online], www.web-based-erp-software.com/, (Access Date Sep 2010)

[14] Welch, J., Kordysh, D. (2007), “Seven Keys to ERP Success.”, Strategic Finance, 89(3), p40-47, 61

[15] Wittmann, A., (2010), “What Cloud Computing Really Means.: Practical Analysis”, InformationWeek, i1263, pp. 40.

[16] Behringer, M.H. (2009), “Classifying Network Complexity”, ACM ReArch’09 Workshop

[17] Force.com, (2009), “A Comprehensive Look at the World’s Premier Cloud-Computing Platform”, Whitepaper, Salesforce, [Online], http://www.developerforce.com/media/Forcedotcom_Whitepaper/WP_Forcedotcom-InDepth_040709_WEB.pdf, (Access Date Jun 2011)

[18] Jagersma, Pieter Klaas, (2004), “Managing Business Complexity”, ManagementSite, [Online], http://www.managementsite.com/461/Managing-Business-Complexity.aspx, (Access Date Sept 2011)

[19] Muketha, G.M. et al, (2010), “a Survey of Business complexity metrics”, Information Technology Journal, 1336-1334.

[20] Sarrell, M. (2010), “Interest Growing in Private Cloud Computing”, eWeek, Ziff Davis Enterprise, [Online], http://www.eweek.com/c/a/Cloud-Computing/Interest-Growing-in-Private-Cloud-Computing-444314/, (Access Date Oct 2010)

[21] Holme, Roberts, Owen, (2008), “The Challenges of SaaS”, BSC, [Presentation] Presentation BSC The Challenges of Saas 080130.pdf

[22] Moody, D.L. (2000), “A Decomposition Method for Entity Relationship Models”, 1st International conference on systems thinking in management

[23] Amaral, L.A.N. and Uzzi, B.(2007), “Complex Systems—A New Paradigm for the Integrative Study of Management, Physical, and Technological Systems”, Management Science Vol. 53, No. 7, July 2007, pp. 1033–1035

[24] Perrone, W.A., Heaphy, M.W., Mahajan, S.D. (2010), "Cloud Computing: Why Forecast Should Matter To You.", Connecticut Law Tribune, [Online], http://www.ctlawtribune.com/getarticle.aspx?ID=38520, (Access Date Oct 2010)

[25] Edmonds, B. (1999), “A Definition of Complexity”, Syntactic Measures of Complexity, University of Manchester, Chapter 4
[26] Levy, D.L. (2000), “Applications and Limitations of Complexity Theory in Organization Theory and Strategy”, Complexity Theory, University of Massachusetts
[27] Wolfram, S. (1985), “Complex Systems Theory”, [Tech. rep.], Institute for Advanced Study, Princeton, NJ 08540 (6-7 October 1984. 1985),
[28] Kearney, J.K. et al., (1986), “Software Complexity Measurement”, Communications of the ACM, Vol. 29 No.11
[29] Simon, H.A. (1973), “The Organisation of Complex Systems”, Hierarchy Theory - The Challenge of Complex Systems
[30] Megiddo, N. (1987), "On the complexity of linear programming," in: Advances in economic theory: Fifth world congress, T. Bewley, ed., Cambridge University Press, Cambridge, 1987, pp. 225–268.
[31] Spiteri, K.J., (2004), “Regulating E-Commerce through Extensible Markup Languages”, Thesis, University of Malta, Malta
[32] Spiteri, K.J., Luca, C. Reynolds, T., Wilson G. (2012), ‘Developing a framework for modelling complex business systems within the cloud’, Proceedings of the International Conference of Information Society (i-Society 2012), London, England, pp. 435 – 440
[33] Geyer, R. and Mackintosh, A. with Lehmann, K. (2005), “What is Complexity Theory?”, Integrating UK and European Social Policy: The Complexity of Europeanisation, Chapt 3
[34] Young, B., Booch, G. et al., (2007), ”Organised and Disorganised Complexity”, Object-Oriented Analysis and Design with Applications
[35] Richardson, K.A. (2004), “Systems Theory and Complexity”, E:CO Issue Vol. 6 No. 4 pp. 77-82
[36] Skyttner, L. (2001), “General Systems Theory: Ideas and Applications”, NJ: World Scientific
[37] Checkland, P. (1981) ,“Systems Thinking, Systems Practice”, Published by John Wiley, page 78
[38] Mikulecky, D.C. (2001), “Definition of Complexity”, Virginia Commonwealth University, [Online], http://views.vce.edu/ mikuleck/. (accessed 2012).
[39] Card, D.N. and Agresti, W.W. (1988), "Measuring Software Design Complexity." The Journal of Systems and Software 8, 3, 185-197.
[40] McCabe, T.J. (1976), “A complexity measure, Software Engineering”, IEEE Transactions on SE-2, no. 4, 308 – 320.
[41] Falgout, J. (2011), “Dataflow Programming: Handling Huge Data Loads Without Adding Complexity”, [Online], http://www.drdobbs.com, (Access date Jun 2012)